

Podstawy Octave

Jacek Golak

Narzędzia obliczeniowe fizyki

UJ WFAIS 2023/2024

<https://octave.org/about>

About

GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. It may also be used as a batch-oriented language.

Octave has extensive tools for solving common numerical linear algebra problems, finding the roots of nonlinear equations, integrating ordinary functions, manipulating polynomials, and integrating ordinary differential and differential-algebraic equations. It is easily extensible and customizable via user-defined functions written in Octave's own language, or using dynamically loaded modules written in C++, C, Fortran, or other languages.

GNU Octave is also freely redistributable software. You may redistribute it and/or modify it under the terms of the [GNU General Public License \(GPL\)](#) as published by the [Free Software Foundation](#).

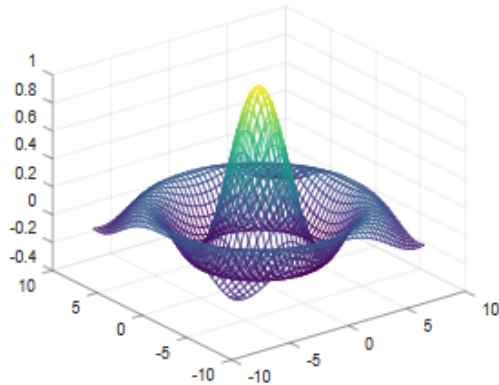
Octave was written by [John W. Eaton](#) and [many others](#). Because Octave is [free software](#) you are encouraged to help make Octave more useful by writing and contributing additional functions for it, and by reporting any problems you may have.

<https://octave.org/index>



Scientific Programming Language

- Powerful mathematics-oriented syntax with built-in 2D/3D plotting and visualization tools
- Free software, runs on GNU/Linux, macOS, BSD, and Microsoft Windows
- Drop-in compatible with many Matlab scripts



⌕ Syntax Examples

The Octave syntax is largely compatible with Matlab. The Octave interpreter can be run in [GUI mode](#), as a console, or invoked as part of a shell script. More Octave examples can be found in [the Octave wiki](#).

Solve systems of equations with linear algebra operations on **vectors** and **matrices**.

```
b = [4; 9; 2] # Column vector
A = [ 3 4 5;
      1 3 1;
      3 5 9 ]
x = A \ b      # Solve the system Ax = b
```



Visualize data with **high-level plot commands** in 2D and 3D.

```
x = -10:0.1:10; # Create an evenly-spaced vector from -10..10
y = sin (x);    # y is also a vector
plot (x, y);
title ("Simple 2-D Plot");
xlabel ("x");
ylabel ("sin (x)");
```

<https://octave.org/support>

Support

Read the **GNU Octave Manual**




-  [Web version](#)
-  [PDF version](#)
- Type `doc` inside Octave

[Octave Wiki](#)

Find [Frequently Asked Questions \(FAQ\)](#)

User community

The Octave user community is a loosely organized association of **volunteers**. Your interactions with the community will be better if you have the [right expectations about the support options](#) available to you.

-  [Discourse](#) – the major communication platform for Octave users and developers.
-  [IRC](#): Chat with users and developers on the [Libera](#) `#octave` [channel](#).
-  Browse the old [mailing list archives](#).

[Commercial support](#)

- Setup assistance
- Custom feature implementation

Polecam następujące strony:

Wiadomości wstępne (Wikipedia)

- http://pl.wikipedia.org/wiki/GNU_Octave
- http://en.wikipedia.org/wiki/GNU_Octave

Dokumentacja Octave

<http://www.gnu.org/software/octave/doc/interpreter> (2018)

<https://octave.org/octave.pdf> (2022)

Przykładowe strony z krótszymi bądź dłuższymi wprowadzeniami do Octave

- https://www.iitis.pl/~miszczak/files/notes/gnu_octave.pdf (2009)
- <http://mst.mimuw.edu.pl/wyklady/ona/wyklad.pdf> (2012)
- <http://www.l5.pk.edu.pl/~mtekieli/files/MS/AMatuszak-Skrypt.pdf> (2010)
- Kursy na YouTube, na przykład <https://www.youtube.com/watch?v=woiU5PRVm7M>

Jak pracować z Octave ?

Octave dostępny jest (za darmo !) dla MS Windows i wielu podstawowych dystrybucji Linuksa.

Z Octave można pracować interakcyjnie (podając po uruchomieniu programu kolejne komendy z klawiatury) lub w trybie „wsadowym” (zapisując kolejne polecenia w pliku tekstowym, a następnie przesyłając cały plik z instrukcjami do wykonania).

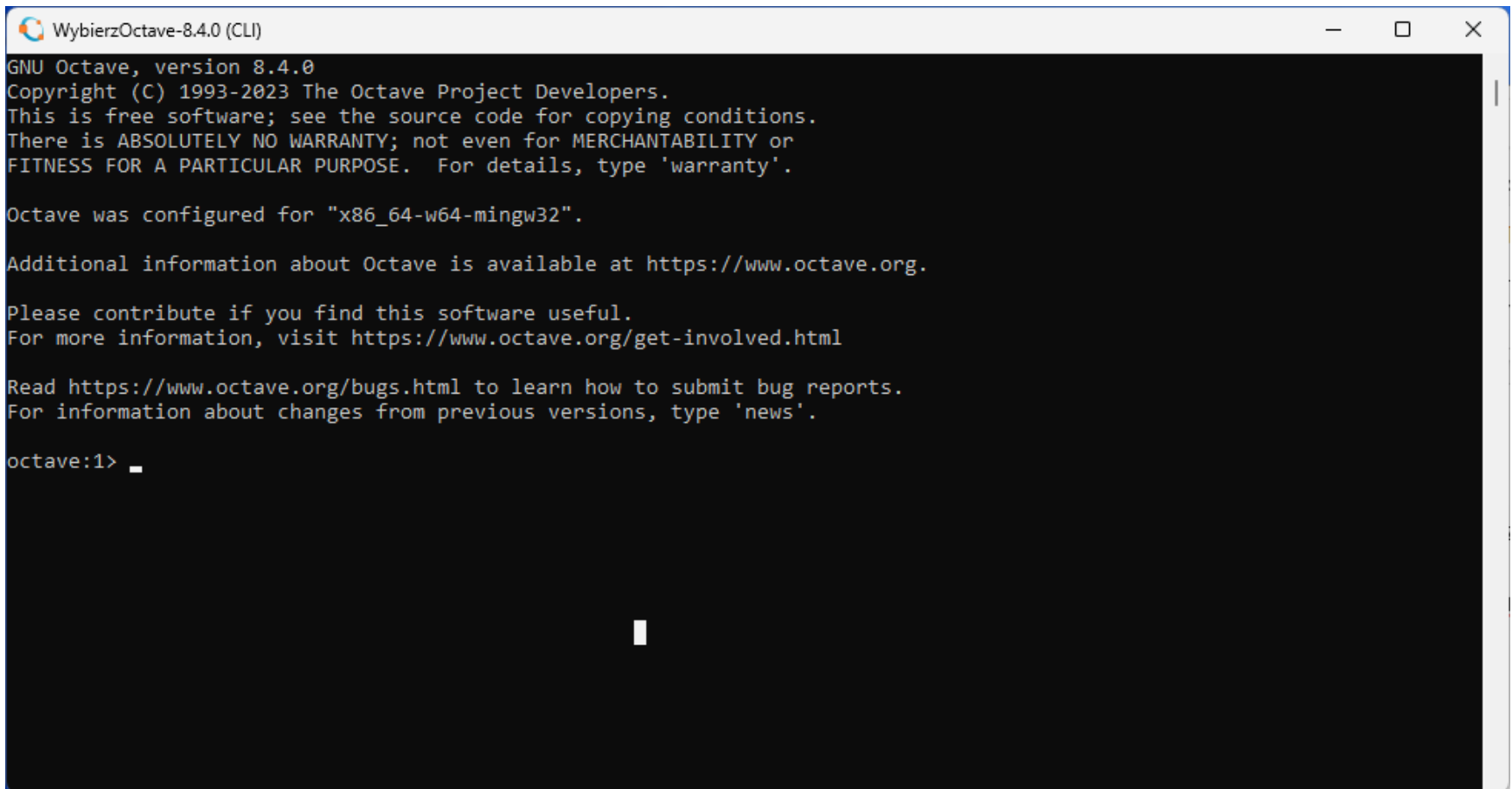
Oba podejścia mają ważne zastosowania.

Na wykładzie ograniczę się do przykładów instrukcji podawanych wprost z linii komend w systemie Linuks.

Zaczynamy (pod Windows) !

Niestety najnowsza wersja GNU Octave ma błąd: przy próbie uruchomienia GNU Octave (GUI) pojawia się także okno (CLI) !

Dlatego pokazuję pracę z linią komend (command line interpreter)



```
WybierzOctave-8.4.0 (CLI)
GNU Octave, version 8.4.0
Copyright (C) 1993-2023 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> _
```

Przykłady ze strony GNU Octave

gnu.octave.7.3.0

```
GNU Octave, version 7.3.0
Copyright (C) 1993-2022 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

>> b = [4; 9; 2] # Column vector
b =

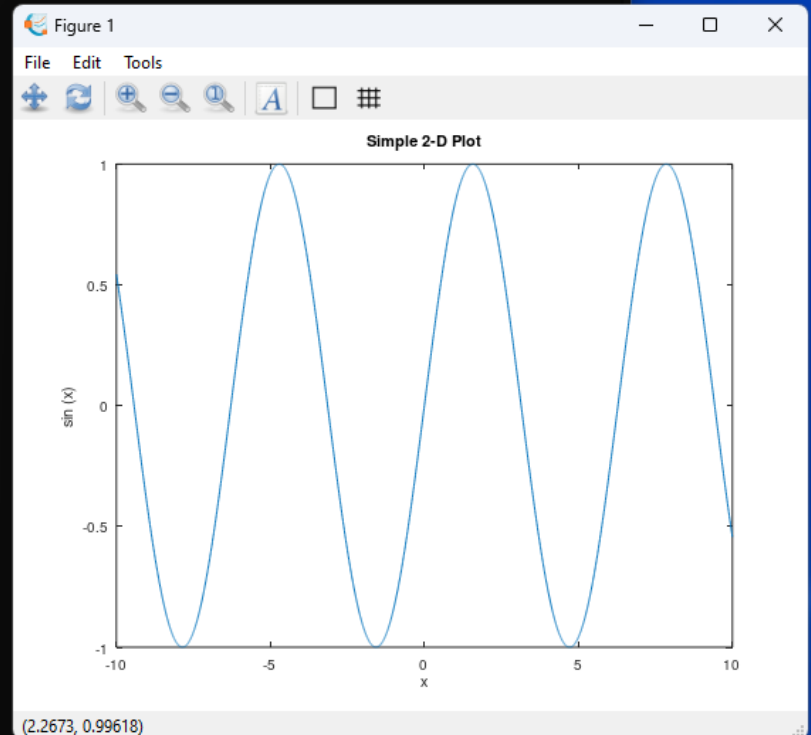
    4
    9
    2

>> A = [ 3 4 5;      1 3 1;      3 5 9 ]
A =

    3    4    5
    1    3    1
    3    5    9

>> x = A \ b      # Solve the system Ax = b
-1.5000
 4.0000
 1.5000

>> #Visualize data with high-level plot commands in 2D and 3D.
>> x = -10:0.01:10; # Create an evenly-spaced vector from -10..10
>> y = sin(x);      # y is also a vector
>> plot(x, y);
>> title("Simple 2-D Plot");
>> xlabel("x");
>> ylabel("sin(x)");
```



Przykłady ze strony GNU Octave

(1) Układ równań $A \cdot x = b$

```
Octave-8.4.0 (CLI)
Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> b = [4; 9; 2] # Column vector
b =

    4
    9
    2

octave:2> A = [ 3 4 5;
>      1 3 1;
>      3 5 9 ]
A =

    3    4    5
    1    3    1
    3    5    9

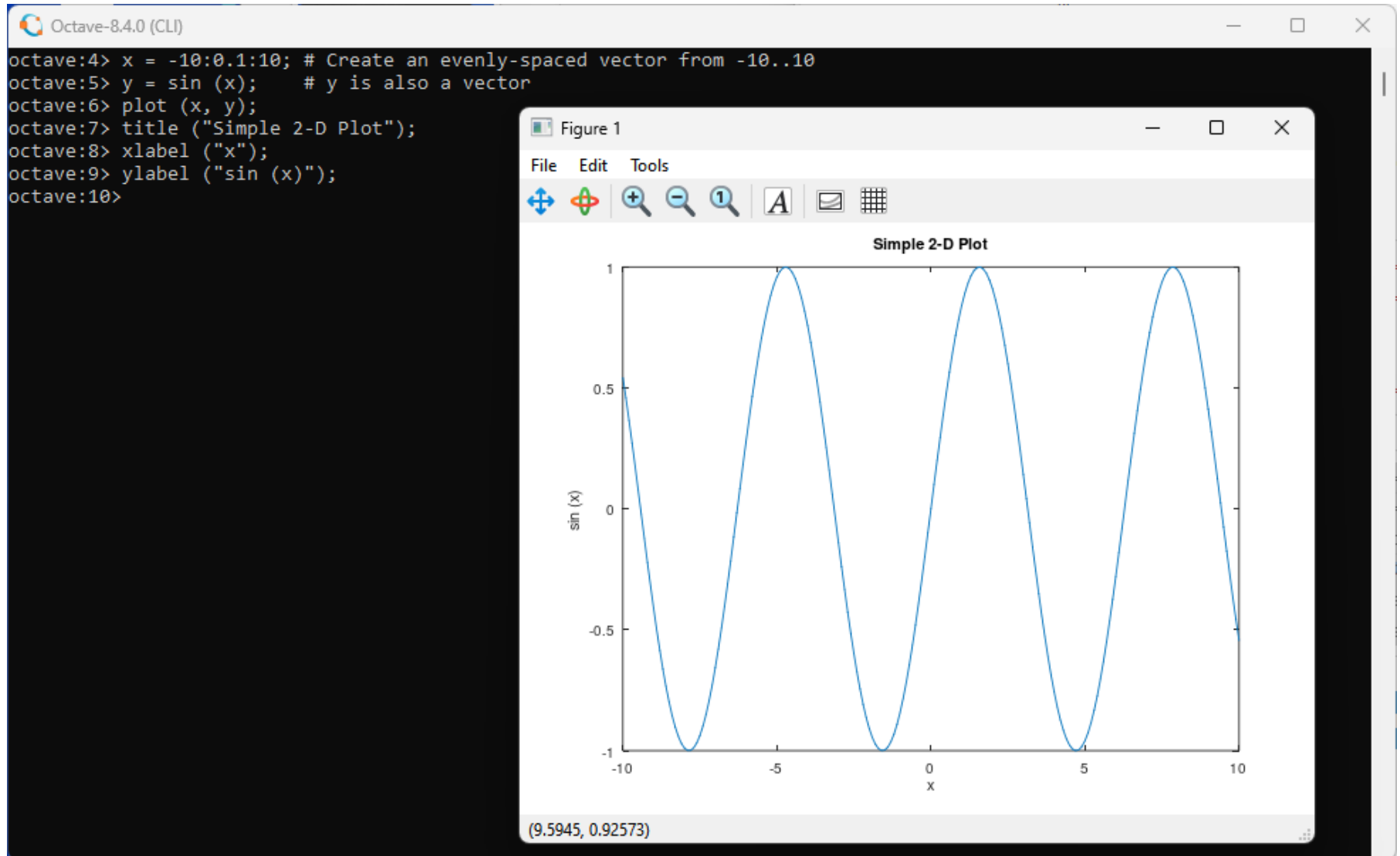
octave:3> x = A \ b      # Solve the system Ax = b
x =

   -1.5000
    4.0000
   -1.5000

octave:4>
```

Przykłady ze strony GNU Octave

(2) Prosty wykres 2D



Przydatne polecenia wstępne:

octave:0> quit % (lub exit) aby zakończyć pracę z Octave

octave:0> pwd % gdzie jestem ?

Octave:0> cd % (z podanym katalogiem docelowym) gdzie chcę się znaleźć

octave:0> ls % (z różnymi opcjami) wyświetla informacje o plikach

octave:0> dir % jak w DOS

octave:0> path % które kartoteki Octave przeszukuje podczas pracy

Octave:0> addpath ("/home/golak/OCTAVE1") % dodanie kartoteki do tych, które są przeszukiwane

Kalkulator

```
octave:0> % OCTAVE jest dobrym naukowym kalkulatorem (to jest komentarz)
```

```
octave:0>
```

```
octave:0> % Na początek tzw. test Knutha: liczymy proste wyrażenie  $(4/3-1)*3-1$ 
```

```
octave:0> (4/3-1)*3-1
```

```
ans = -2.2204e-16
```

```
octave:1> % Co otrzymalibyśmy w przypadku programu MATHEMATICA ?
```

```
octave:1> % Dlaczego ?
```

```
octave:1>
```

```
octave:1> % Jak wypisywać wyniki w sposób
```

```
octave:1> % bardziej kontrolowany (disp i printf) ?
```

```
octave:1>
```

```
octave:1> disp(" sin(2.)= "),disp(sin(2.));
```

```
sin(2.)=
```

```
0.90930
```

```
octave:1.2>
```

```
octave:1.2> printf(" sin(2.)= %f\n ",sin(2.)); % funkcja z języka C
```

```
sin(2.)= 0.909297
```

Kalkulator cd.

```
octave:1.3>
```

```
octave:1.3> % OCTAVE zna bardzo wiele różnych funkcji
```

```
octave:1.3> 1/3 + 4/3 + 2.5*sin(log(11+cos(234))) + tan(-27)/asin(1/9)
```

```
ans = 32.754
```

```
octave:2>
```

```
octave:2> % Wyrażenie może być wartością zmiennej
```

```
octave:2> % Uwaga: OCTAVE rozróżnia małe i duże litery
```

```
octave:2> x = 1/3 + 4/3 + 2.5*sin(log(11+cos(234))) + tan(-27)/asin(1/9)
```

```
x = 32.754
```

```
octave:3>
```

```
octave:3> % Wynik nie jest wyświetlany, gdy na końcu damy średnik
```

```
octave:3> x = 1/3 + 4/3 + 2.5*sin(log(11+cos(234))) + tan(-27)/asin(1/9) ;
```

```
octave:4>
```

```
octave:4> y = x^(sin(1300)-exp(2.4))
```

```
y = 2.6146e-18
```

Wartości specjalne

octave:5> % Wartości specjalne do obsługi błędów numerycznych

octave:5> 1/0

warning: division by zero

ans = Inf

octave:6> -1/0

warning: division by zero

ans = -Inf

octave:7> 1/0-2/0

warning: division by zero

warning: division by zero

ans = NaN

octave:8>

octave:8> % Największa liczba zmiennoprzecinkowa

octave:8> realmax

ans = 1.7977e+308

octave:9> % Najmniejsza co do modułu liczba zmiennoprzecinkowa

octave:9> realmin

ans = 2.2251e-308

octave:10> % Parametr dokładności numerycznej

octave:10> eps

ans = 2.2204e-16

Niektóre stałe

```
octave:11> % WAŻNE STAŁE MATEMATYCZNE
```

```
octave:11>
```

```
octave:11> % podwojone pierwsze dodatnie miejsce zerowe funkcji cosinus,
```

```
octave:11> % czyli po prostu pi=3.1415...
```

```
octave:11> pi
```

```
ans = 3.1416
```

```
octave:12> % podstawa logarytmu naturalnego e=2.71...
```

```
octave:12> e
```

```
ans = 2.7183
```

```
octave:13> % jednostka urojona i ( $i*i=-1$ ); także l,j,J
```

```
octave:13> i
```

```
ans = 0 + 1i
```

```
octave:14> l
```

```
ans = 0 + 1i
```

```
octave:15> j
```

```
ans = 0 + 1i
```

```
octave:16> J
```

```
ans = 0 + 1i
```

```
octave:17> i*i
```

```
ans = -1
```

Liczby zespolone

octave:18> % OCTAVE radzi sobie z liczbami zespolonymi

octave:18> z1= 1+i;

octave:19> z2=1-i;

octave:20> z3=z1/z2

z3 = 0 + 1i

octave:21>

octave:21> % moduł liczby zespolonej

octave:21> abs(z3)

ans = 1

octave:22>

octave:22> % sprzężenie liczby zespolonej

octave:22> % conj(2-3*j)

ans = 2 + 3i

octave:23>

octave:23> % faza (inaczej argument) liczby zespolonej

octave:23> arg(z3)

ans = 1.5708

octave:24>

octave:24> % Ale uwaga na konwencje !

octave:24> arg(-1-j) % w radianach w przedziale [-pi, pi]

ans = -2.3562

Macierze (chyba najważniejsza część Octave)

```
octave:25> % Macierze: uwaga na składnię
```

```
octave:25> % Wektor kolumnowy o czterech elementach
```

```
octave:25> v1=[ 1 ; 2 ; 3 ; 4]
```

```
v1 =
```

```
1
```

```
2
```

```
3
```

```
4
```

```
octave:26> % Wektor wierszowy o czterech elementach
```

```
octave:26> v2= [5,6,7,8]
```

```
v2 =
```

```
5.0000 6.0000 7.8000 8.0000
```

```
octave:27> v2= [5,6,7,8]
```

```
v2 =
```

```
5 6 7 8
```

Macierze cd.

```
octave:28> % Macierz 4 x 4
```

```
octave:28> A=[9,10,11,12;13,14,15,16;17,18,19,20;21,22,23,24]
```

```
A =
```

```
 9  10  11  12
13  14  15  16
17  18  19  20
21  22  23  24
```

```
octave:29> B=[9-i,10+i,11-2*i,12-i;13+3*i,14,15-2*i,16-i;17+i,18-i,19-
3*i,20;21,22+2*i,23,24]
```

```
B =
```

```
9 - 1i  10 + 1i  11 - 2i  12 - 1i
13 + 3i  14 + 0i  15 - 2i  16 - 1i
17 + 1i  18 - 1i  19 - 3i  20 + 0i
21 + 0i  22 + 2i  23 + 0i  24 + 0i
```

Macierze cd.

```
octave:30> % Mnożenie macierzowe A v1
```

```
octave:30> A*v1
```

```
ans =
```

```
110
```

```
150
```

```
190
```

```
230
```

```
octave:31> % Mnożenie macierzowe v2 A
```

```
octave:31> v2*A
```

```
ans =
```

```
410 436 462 488
```

```
octave:32> % Mnożenie macierzowe v2 v1
```

```
octave:32> % Uwaga: macierz 1 x 1 utożsamiana jest z liczbą
```

```
octave:32> v2*v1
```

```
ans = 70
```

Macierze cd.

```
octave:33> % Mnożenie macierzowe v1 v2
```

```
octave:33> v1*v2
```

```
ans =
```

5	6	7	8
10	12	14	16
15	18	21	24
20	24	28	32

Macierze cd.

```
octave:34> % Przypomnijmy sobie macierz A
```

```
octave:34> A
```

```
A =
```

```
 9 10 11 12
13 14 15 16
17 18 19 20
21 22 23 24
```

```
octave:35> % Prim (') oznacza transponowanie dla macierzy rzeczywistej
```

```
octave:35> A'
```

```
ans =
```

```
 9 13 17 21
10 14 18 22
11 15 19 23
12 16 20 24
```

Macierze cd.

```
octave:36> % Przypomnijmy sobie macierz B
```

```
octave:36> B
```

```
B =
```

```
9 - 1i 10 + 1i 11 - 2i 12 - 1i  
13 + 3i 14 + 0i 15 - 2i 16 - 1i  
17 + 1i 18 - 1i 19 - 3i 20 + 0i  
21 + 0i 22 + 2i 23 + 0i 24 + 0i
```

```
octave:37> % Dla zespolonej macierzy prim (') oznacza sprzężenie hermitowskie
```

```
octave:37> % czyli sprzężenie zespolone wszystkich elementów oraz zamiana
```

```
octave:37> % wierszy z kolumnami (transponowanie)
```

```
octave:37> B'
```

```
ans =
```

```
9 + 1i 13 - 3i 17 - 1i 21 - 0i  
10 - 1i 14 - 0i 18 + 1i 22 - 2i  
11 + 2i 15 + 2i 19 + 3i 23 - 0i  
12 + 1i 16 + 1i 20 - 0i 24 - 0i
```

Macierze cd.

```
octave:38> % Dodanie kropki przed prim (') daje czyste transponowanie
```

```
octave:38> % macierzy B bez sprzężenia zespolonego
```

```
octave:38> B.'
```

```
ans =
```

```
9 - 1i 13 + 3i 17 + 1i 21 + 0i  
10 + 1i 14 + 0i 18 - 1i 22 + 2i  
11 - 2i 15 - 2i 19 - 3i 23 + 0i  
12 - 1i 16 - 1i 20 + 0i 24 + 0i
```

Macierze cd.

octave:39> % Macierze można dodawać, jeśli ich rozmiary są zgodne

octave:39> A+B

ans =

18 - 1i	20 + 1i	22 - 2i	24 - 1i
26 + 3i	28 + 0i	30 - 2i	32 - 1i
34 + 1i	36 - 1i	38 - 3i	40 + 0i
42 + 0i	44 + 2i	46 + 0i	48 + 0i

octave:40> % Można mnożyć całą macierz przez liczbę

octave:40> 3*A

ans =

27	30	33	36
39	42	45	48
51	54	57	60
63	66	69	72

Macierze cd.

```
octave:41> % Przykład kombinacji liniowej macierzy A i B
```

```
octave:41> i*A+2*B
```

```
ans =
```

```
18 + 7i 20 + 12i 22 + 7i 24 + 10i  
26 + 19i 28 + 14i 30 + 11i 32 + 14i  
34 + 19i 36 + 16i 38 + 13i 40 + 20i  
42 + 21i 44 + 26i 46 + 23i 48 + 24i
```

```
octave:42> % KAŻDY element macierzy A zwiększamy o 4
```

```
octave:42> A+4
```

```
ans =
```

```
13 14 15 16  
17 18 19 20  
21 22 23 24  
25 26 27 28
```

Macierze cd.

```
octave:42.2> % Można zadziałać funkcją na macierz, np. sin(A).
```

```
octave:42.2> % UWAGA: oznacza to, że funkcja sinus działa
```

```
octave:42.2> % z osobna na KAŻDY element macierzy A.
```

```
octave:42.2> % Nie jest to właściwy sinus macierzy !
```

```
octave:42.2> sin(A)
```

```
ans =
```

```
0.4121185 -0.5440211 -0.9999902 -0.5365729
```

```
0.4201670 0.9906074 0.6502878 -0.2879033
```

```
-0.9613975 -0.7509872 0.1498772 0.9129453
```

```
0.8366556 -0.0088513 -0.8462204 -0.9055784
```

```
octave:42.3> % Elementy diagonalne macierzy
```

```
octave:42.3> diag(A)
```

```
ans =
```

```
9
```

```
14
```

```
19
```

```
24
```

Macierze cd.

```
octave:42.4> % Wyznacznik macierzy kwadratowej
```

```
octave:42.4> det(A)
```

```
ans = -1.1833e-29
```

```
octave:42.5> det(B)
```

```
ans = 200.00 - 490.00i
```

```
octave:42.6> % macierz odwrotna (jeśli istnieje)
```

```
octave:42.6> inverse(A) % nie istnieje macierz odwrotna do macierzy A
```

```
warning: inverse: matrix singular to machine precision, rcond = 2.05597e-18
```

```
ans =
```

```
3.1250e-01  5.6295e+14 -1.1259e+15  5.6295e+14  
-6.0048e+14 -5.0665e+14  2.8147e+15 -1.7076e+15  
1.2010e+15 -6.7554e+14 -2.2518e+15  1.7264e+15  
-6.0048e+14  6.1924e+14  5.6295e+14 -5.8171e+14
```

Macierze cd.

octave:42.7> inverse(B) % tu nie powinno być problemu

ans =

Columns 1 through 3:

-0.019065 + 0.083292i	-0.035487 - 0.256944i	0.030561 + 0.054873i
-0.090396 - 0.481471i	-0.213210 + 0.017637i	0.176366 + 0.132096i
0.088754 + 0.447447i	0.225134 - 0.118422i	-0.184220 + 0.248661i
-0.025634 - 0.052803i	0.012210 + 0.339914i	-0.000857 - 0.422099i

Column 4:

0.014959 + 0.081649i
0.059693 + 0.106248i
-0.054659 - 0.338915i
0.035095 + 0.150982i

Macierze cd.

```
octave:42.8> % sprawdzenie B^(-1) B
```

```
octave:42.8> inverse(B)*B
```

```
ans =
```

Columns 1 through 3:

1.00000 + 0.00000i	0.00000 - 0.00000i	0.00000 + 0.00000i
-0.00000 - 0.00000i	1.00000 - 0.00000i	0.00000 + 0.00000i
-0.00000 - 0.00000i	-0.00000 - 0.00000i	1.00000 + 0.00000i
-0.00000 + 0.00000i	-0.00000 + 0.00000i	0.00000 - 0.00000i

Column 4:

0.00000 - 0.00000i
0.00000 - 0.00000i
-0.00000 - 0.00000i
1.00000 - 0.00000i

Macierze cd.

```
octave:42.9> % sprawdzenie B B^(-1)
```

```
octave:42.9> B*inverse(B)
```

```
ans =
```

Columns 1 through 3:

```
1.00000 - 0.00000i -0.00000 + 0.00000i -0.00000 + 0.00000i
0.00000 + 0.00000i 1.00000 + 0.00000i -0.00000 - 0.00000i
0.00000 + 0.00000i -0.00000 + 0.00000i 1.00000 + 0.00000i
0.00000 + 0.00000i -0.00000 + 0.00000i -0.00000 + 0.00000i
```

Column 4:

```
0.00000 + 0.00000i
0.00000 + 0.00000i
0.00000 + 0.00000i
1.00000 + 0.00000i
```

Macierze cd.

```
octave:43> ones(3,4)
```

```
ans =
```

```
1 1 1 1
1 1 1 1
1 1 1 1
```

```
octave:44> zeros(3,4)
```

```
ans =
```

```
0 0 0 0
0 0 0 0
0 0 0 0
```

```
octave:45> eye(3,4)
```

```
ans =
```

Diagonal Matrix

```
1 0 0 0
0 1 0 0
0 0 1 0
```

Macierze cd.

```
octave:46> % Użyteczne operacje na macierzach
```

```
octave:46>
```

```
octave:46> rows(A) % ile wierszy ma macierz A
```

```
ans = 4
```

```
octave:47> columns(A) % ile kolumn ma macierz B
```

```
ans = 4
```

```
octave:48> % albo po prostu pytamy o rozmiar macierzy
```

```
octave:48> size(A)
```

```
ans =
```

```
4 4
```

```
octave:49> % Wyciągamy k-tą kolumnę z macierzy: A(:,k)
```

```
octave:49> v3=A(:,2)
```

```
v3 =
```

```
10
```

```
14
```

```
18
```

```
22
```


Macierze cd.

```
octave:50> % Wyciągamy k-tą wiersz z macierzy: A(k,:)
```

```
octave:50> A(3,:)
```

```
ans =
```

```
17 18 19 20
```

```
octave:51> % Dla macierzy kwadratowej możemy policzyć ślad
```

```
octave:51> % czyli sumę elementów na głównej przekątnej
```

```
trace(A)
```

```
octave:51> trace(B*B' + B'*B)
```

```
ans = 9464
```

Macierze cd.

```
octave:52> % największy element wektora lub wiersz macierzy  
octave:52> % zawierający jej największy element  
octave:52> max(A)  
ans =
```

21 22 23 24

```
octave:53> % najmniejszy element wektora lub wiersz macierzy  
octave:53> % zawierający jej najmniejszy element  
octave:53> min(A)  
ans =
```

9 10 11 12

Problem własny

```
octave:54> % Problem własny: wartości i wektory własne macierzy
```

```
octave:54> % na przykładzie macierzy A
```

```
octave:54> A
```

```
A =
```

```
 9 10 11 12
13 14 15 16
17 18 19 20
21 22 23 24
```

```
octave:55> % Typowe użycie funkcji eig:
```

```
octave:55> % Wektory własne to kolumny macierzy V
```

```
octave:55> % wartości własne to elementy diagonalne macierzy lambda
```

```
octave:55> [V,lambda]=eig(A)
```

Problem własny cd.

V =

Columns 1 through 3:

-0.31058 + 0.00000i	0.70255 + 0.00000i	-0.05587 - 0.10301i
-0.42564 + 0.00000i	0.25632 + 0.00000i	-0.32394 + 0.15451i
-0.54070 + 0.00000i	-0.18991 + 0.00000i	0.81547 + 0.00000i
-0.65575 + 0.00000i	-0.63613 + 0.00000i	-0.43567 - 0.05150i

Column 4:

-0.05587 + 0.10301i
-0.32394 - 0.15451i
0.81547 - 0.00000i
-0.43567 + 0.05150i

Problem własny cd.

lambda =

Diagonal Matrix

Columns 1 through 3:

67.19064 + 0.00000i		0	0
0	-1.19064 + 0.00000i		0
0	0	0	-0.00000 + 0.00000i
0	0	0	0

Column 4:

0
0
0
-0.00000 - 0.00000i

Problem własny cd.

```
octave:56> % Teraz sprawdzimy, czy mamy rzeczywiście wartości
```

```
octave:56> % i wektory własne A !
```

```
octave:56> l1=lambda(1,1)
```

```
l1 = 67.191
```

```
octave:57> v1=V(:,1)
```

```
v1 =
```

```
-0.31058
```

```
-0.42564
```

```
-0.54070
```

```
-0.65575
```

```
octave:58> chk1=A*v1-l1*v1
```

```
chk1 =
```

```
1.4211e-14
```

```
7.1054e-15
```

```
1.4211e-14
```

```
7.1054e-15
```

Problem własny cd.

```
octave:59> l2=lambda(2,2)
```

```
l2 = -1.1906
```

```
octave:60> v2=V(:,2)
```

```
v2 =
```

```
0.70255
```

```
0.25632
```

```
-0.18991
```

```
-0.63613
```

```
octave:61> chk2=A*v2-l2*v2
```

```
chk2 =
```

```
8.8818e-16
```

```
-1.2212e-15
```

```
1.6653e-15
```

```
-3.1086e-15
```

Problem własny cd.

```
octave:62> l3=lambda(3,3)
```

```
l3 = 1.1971e-16 + 4.5219e-16i
```

```
octave:63> v3=V(:,3)
```

```
v3 =
```

```
-0.09491 - 0.07195i
```

```
-0.27176 + 0.10792i
```

```
0.82824 + 0.00000i
```

```
-0.46157 - 0.03597i
```

```
octave:64> chk3=A*v3-l3*v3
```

```
chk3 =
```

```
-9.0935e-16 - 2.2603e-16i
```

```
8.1333e-17 - 4.4515e-16i
```

```
-1.8755e-15 - 1.0407e-15i
```

```
3.8990e-17 - 5.6413e-16i
```


Problem własny cd.

```
octave:65> chk4=A*V(:,4)-lambda(4,4)*V(:,4)  
chk4 =
```

```
-9.0935e-16 + 2.2603e-16i  
8.1333e-17 + 4.4515e-16i  
-1.8755e-15 + 1.0407e-15i  
3.8990e-17 + 5.6413e-16i
```

Definiowanie nowych funkcji

```
octave:66> % Definiowanie własnych funkcji
octave:66>
octave:66> % Na początek funkcja jednej zmiennej (zespólonej)
octave:66> function y=f1(x)
> y=x*x + 1 ;
> endfunction
octave:67> f1(2)
ans = 5
octave:68> f1(2+3*i)
ans = -4 + 12i
octave:69>
octave:69> % Ten sam efekt można uzyskać, przygotowując
octave:69> % wcześniej osobny plik z definicją funkcji.
octave:69> % Jego nazwa musi zgadzać się z nazwą funkcji i mieć
octave:69> % rozszerzenie "*.m". Można taki pliki lub takie pliki
octave:69> % umieścić w bieżącej kartotece lub w jednej z kartotek,
octave:69> % które OCTAVE przeszukuje.
```

Definiowanie nowych funkcji cd.

```
octave:69> % Sprawdzam zawartość pliku "f2.m"
octave:69> system("cat f2.m");
function y=f2(x)
y=x*x + 1 ;
endfunction
octave:70>
octave:70> % f2 działa tak samo jak f1
octave:70> f2(2)
ans = 5
octave:71> f2(2+3*i)
ans = -4 + 12i
octave:72>
octave:72> % Teraz funkcja trzech zmiennych (zespolonych)
octave:72> function y=f3(x1,x2,x3)
> y=x1+x2-2*x3 ;
> endfunction
octave:73>
octave:73> f3(1,-i,17)
ans = -33 - 1i
```

Rysowanie wykresów

octave:74> % Rysowanie wykresów funkcji, używając funkcji "plot":

octave:74> % plot(x,y) rysuje linie łączące punkty (xi,yi),

octave:74> % gdzie xi (yi) to składowe wektora x (y).

octave:74> % Wektory x i y mają tę samą długość:

octave:74> % Przykład nr 1

octave:75> x=[0:1:10]

x =

0 1 2 3 4 5 6 7 8 9 10

octave:75> y=sin(x) % wektor wartości sin(xi)

y =

Columns 1 through 8:

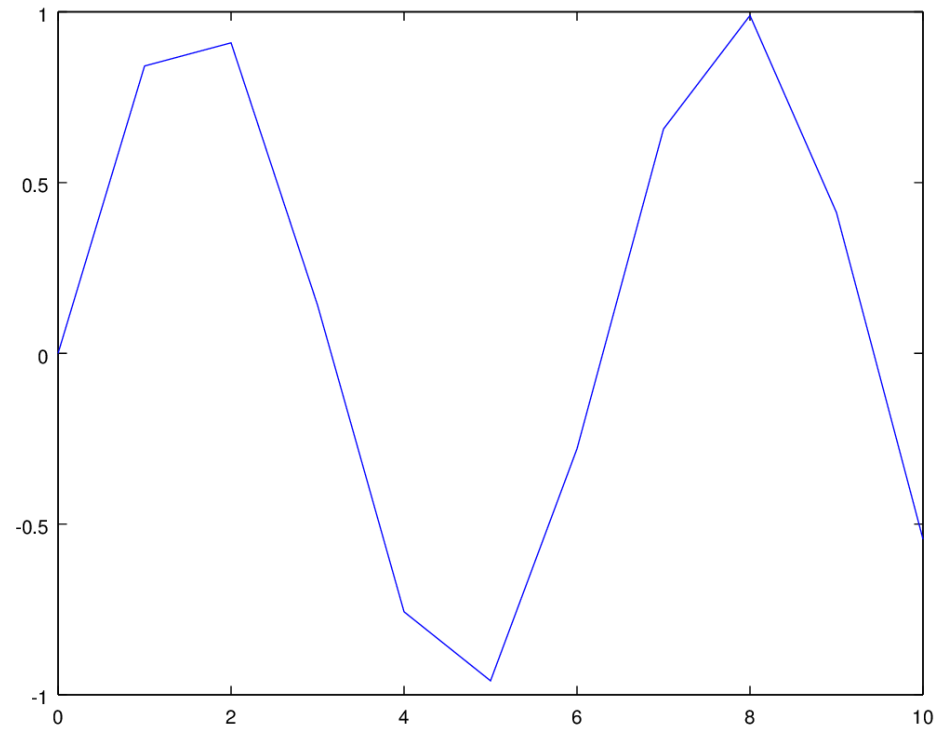
0.00000 0.84147 0.90930 0.14112 -0.75680 -0.95892 -0.27942 0.65699

Columns 9 through 11:

0.98936 0.41212 -0.54402

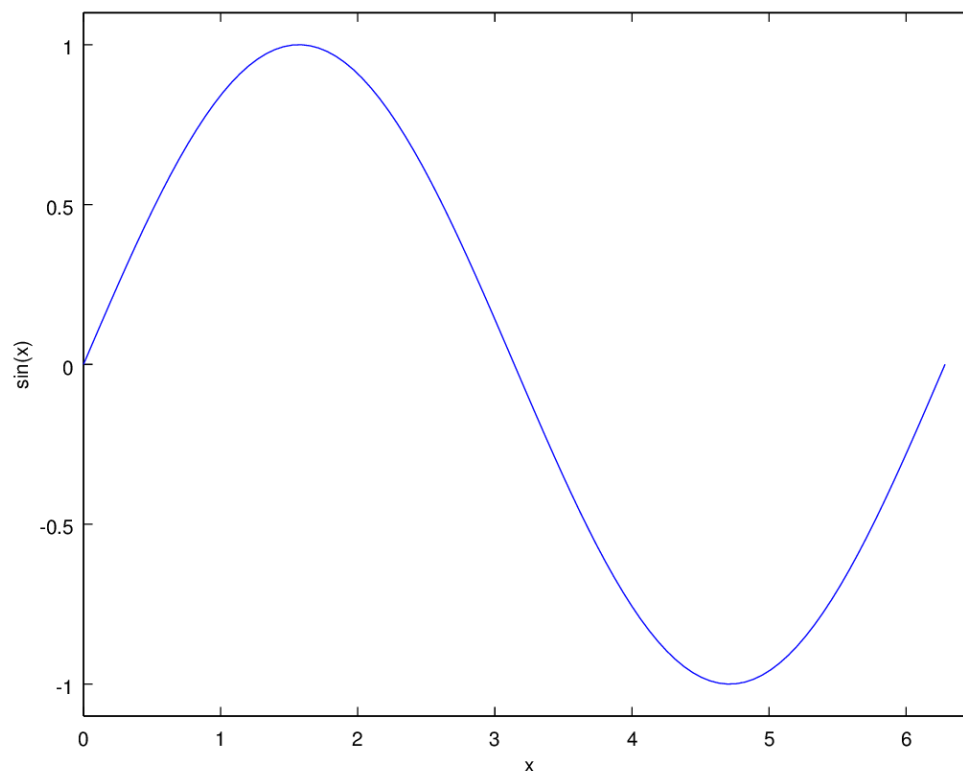
Rysowanie wykresów cd.

```
octave:76> plot(x,y)
```



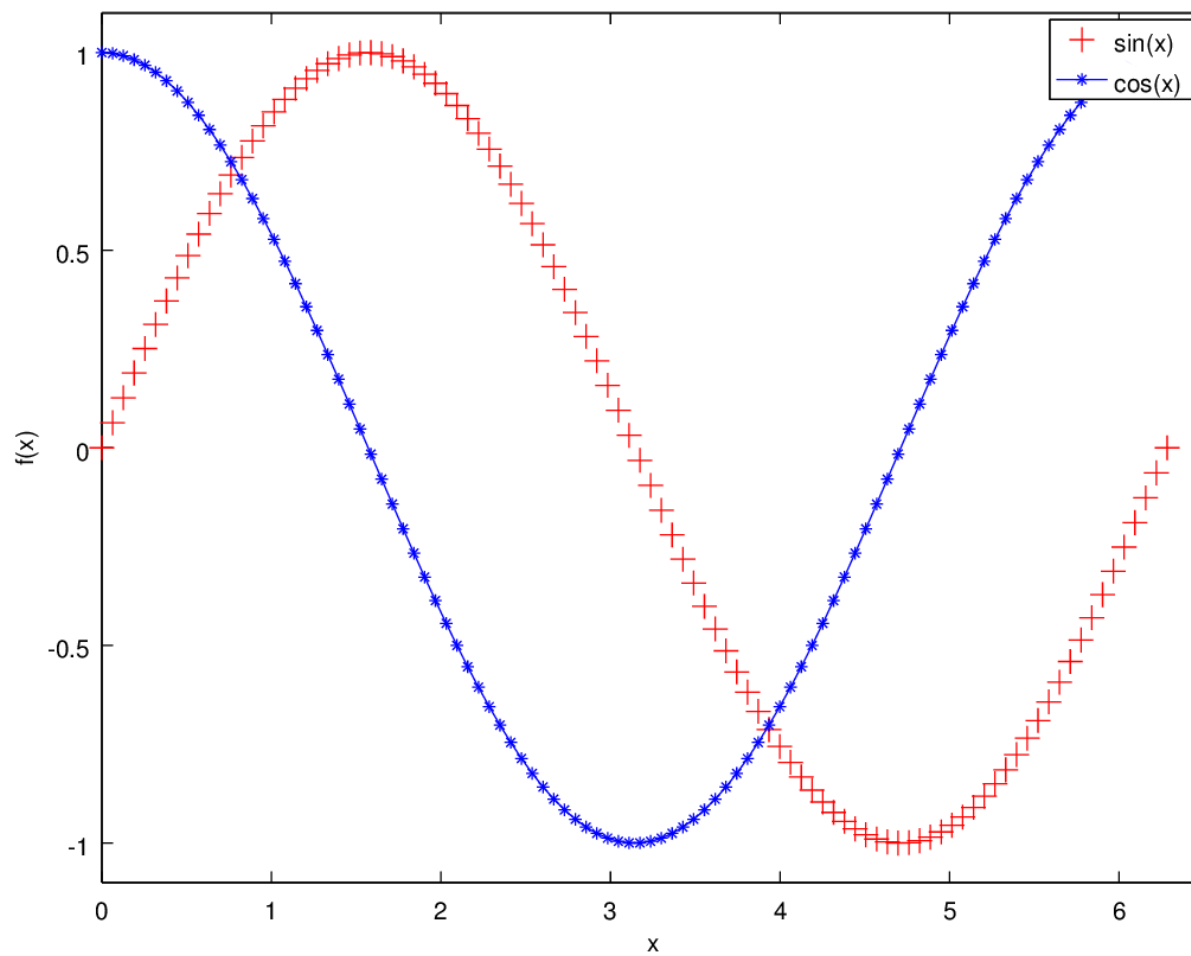
Rysowanie wykresów cd.

```
octave:76.1> % Przykład nr 2  
octave:76.1> x=linspace(0, 2*pi, 100);  
octave:77> y = sin(x);  
octave:78> plot(x,y)  
octave:79> axis( [0 6.5 -1.1 1.1] )  
octave:80> xlabel('x')  
octave:81> ylabel('sin(x)')
```



Rysowanie wykresów cd.

```
octave:82> % Przykład nr 3: więcej linii
octave:82> x = linspace(0, 2*pi, 100);
octave:83> y1 = sin(x);
octave:84> y2 = cos(x);
octave:85> plot(x, y1, '+1;sin(x);', "markersize", 10, x, y2, ";cos(x);", "markersize", 5,
"marker", '*')
octave:86> axis( [0 6.5 -1.1 1.1] )
octave:87> xlabel('x')
octave:88> ylabel('f(x)')
octave:89> % Można zapisać rysunek
Octave:89> % w różnych formatach, np. png
octave:89> print -dpng przyklad3.png % może lepiej z podaniem ścieżki
Octave:89.1> print -dpng C:\Users\JacekG\Desktop\przyklad3.png
```



Rysowanie wykresów cd.

octave:90> % Uwagi:

octave:90> % (1) możliwe są inne rodzaje wykresów 2D: semilogx, semilogy, loglog, polar

octave:90> % (2) możliwe są wykresy 3D: mesh, meshgrid, surf

Rozwiązywanie równań

```
octave:90> % Rozwiązywanie równań
octave:90> % Przykład: szukamy x w pobliżu 1, dla którego  $\sin(x)-\cos(x)=0$ 
octave:90> % Wymaga to zdefiniowania funkcji (f3), która będzie parametrem
octave:90> % funkcji fsolve. Drugim parametrem fsolve jest punkt startowy,
octave:90> % czyli przybliżone rozwiązanie
octave:90> function y=f3(x)
> y=sin(x)-cos(x);
> endfunction
octave:91> x0=fsolve("f3",1.0)
x0 = 0.78540
```

Całkowanie numeryczne

```
octave:92> % całkowanie numeryczne przy pomocy funkcji quad
```

```
octave:92> quad("f3",0,pi)
```

```
ans = 0.50685
```

```
octave:93> % Funkcja f3 ma oczywiście pierwotną, -sin(x)-cos(x),
```

```
octave:93> % więc ten wynik możemy łatwo sprawdzić
```

```
octave:93> (-sin(2)-cos(2)) - (-sin(0)-cos(0))
```

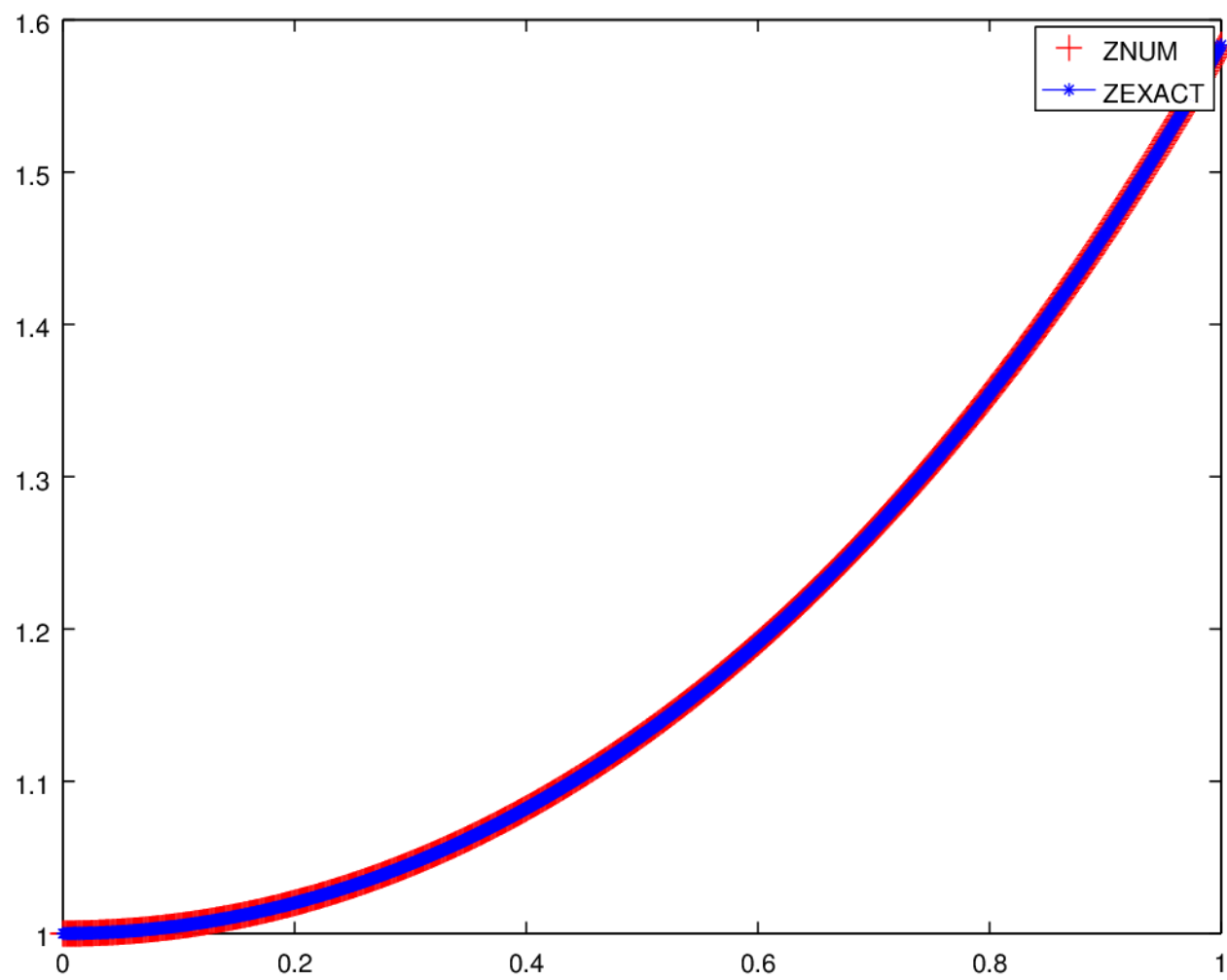
```
ans = 0.50685
```

Równanie różniczkowe

```
octave:94> % Pojedyncze (nieuwikłane) równanie różniczkowe I rzędu
octave:94> %  $y'(x)=y(x)*\sin(x)$ ,  $y(0)=1$ ,  $0 \leq x \leq 1$ . (*)
octave:94> % To równanie ma rozwiązanie w postaci
octave:94> %  $y(x) = \exp(1-\cos(x))$ 
octave:94> % W OCTAVE najpierw definiujemy funkcję f4(x,t), która
octave:94> % jest prawą stroną równania (*):
octave:94> function y=f4(x,t)
> y=x*sin(t);
> endfunction
```

Równanie różniczkowe cd.

```
octave:95> % zadajemy przedział, na którym budujemy rozwiązanie
octave:95> t=[0:0.001:1];
octave:96> % zadajemy warunek początkowy
octave:96> x0=1;
octave:97> % numeryczne rozwiązanie z użyciem funkcji lsode
octave:97> Z=lsode("f4", x0, t);
octave:98> % dokładne (analityczne) rozwiązanie w tych samych punktach t
octave:98> ZEXACT=exp(1-cos(t));
octave:99> plot(t, Z, '+1;ZNUM;', "markersize", 10, t, ZEXACT, ";ZEXACT;",
"markersize", 5, "marker", '*')
octave:100> print -dpng C:\Users\JacekG\Desktop\przyklad_lsode.png
```



Równanie różniczkowe cd.

octave:101> % Uwagi:

octave:101> % (1) Równanie różniczkowe wyższego niż I rzędu zapisujemy

octave:101> % jako układ równań pierwszego rzędu

octave:101> % Taki układ jawnych równań też może być rozwiązany przy pomocy
lsode

octave:101> % (2) W wielu przypadkach konieczne są dodatkowe opcje, by uzyskać

octave:101> % poprawne rozwiązanie.

Programowanie w Octave

Octave umożliwia pisanie programów, podobnie jak to jest na przykład w języku C lub Fortranie. Oznacza to, że oprócz instrukcji podstawiania mamy do dyspozycji instrukcje warunkowe, pętle itd.

Do Octave można „włożyć” kawałki kodu przygotowane w C lub w Fortranie.

Tymi zagadnieniami nie będziemy się zajmować w tym wprowadzeniu do Octave.

Reszta jest ćwiczeniem !